[JavaScript 参考 - JavaScript | MDN (mozilla.org)]

构造函数

构造函数：是对事物的抽象（类）；体现在JS语法上：

1.就是函数；

2.建议首字母大写；

调用方式：普通函数直接调用；构造函数使用new关键字调用

```
function Person(){
    console.log(this); // Person{}
}
// 普通函数
var fun = Person();
console.log(fun); // undefined
// 构造函数
var person = new Person();
console.log(person); // Person{}
```

3.构造函数是用来创建实例化对象的，所以需要有个东西指向构造函数创建的实例化对象：this；

4.构造函数里面的this指向的是构造函数创建的实例化对象；

5.构造函数默认返回值是this（一般我们就是用这个默认值）；

6.new Person(pname, gender); 就是创建一个实例化对象；必须用new来调用构造函数

构造函数的执行流程:

- 立刻创建一个新的对象
- 将新建的对象设置为函数中this，在构造函数中可以使用this来引用新建的对象
- 逐行执行函数中的代码
- 将新建的对象作为返回值返回（不需要return返回结果）

this的情况:

- 当以函数的形式调用时，this是window
- 当以方法的形式调用时，谁调用方法this就是谁
- 当以构造函数的形式调用时，this就是新创建的那个对象

```
function  Person(name, age, gender){
  /** 函数的形参就是函数内部的全局变量*/
 console.log(name);
   //1，在构造函数内部，给构造函数创建的实例化对象设置特征（属性）或者动作（方法）；
   //2，使用this指向（代表，表示）当前构造函数创建的实例化对象；
```

```
    //3，通过this往实例化对象身上追加属性或方法；

   this.name = name;  // ====>  this['pname']
   //习惯性把属性名和形参保持一致；使用形参对属性进行初始化；后续操作会对属性的值进行修改
   this.age = age; //====> this['age']
   this.say= function(){  //通过属性获取对象的相关数据
        retrun `我是$(this.name),今年${this.age}岁，是一名${this.gender}`
   }
  }
   //使用new创建构造函数的实例化对象,也就是用new关键字调用函数
  var p1 = new Person('ABC', 19);
  console.log(p1);

var p2 = new Person('小名', 20);
console.log(p2);

//考察this指向和原型链
   function obj(name) {
            console.log(this);
            if (this === window) { //判断通过何种方式调用。
                if (name) {
                    this.name = name;
                } else {
                    this.name = 'name1';
                }
                return this;
            }
        }
        obj.prototype.name = "name2";
        var a = obj("name1"); //如果通过函数方式调用,this会指向window。
        var b = new obj(); //如果通过new方式调用,this会指向实例化后的对象,obj{}
        var c = obj(); //如果 函数调用的时候不带参数,默认name为name1
        console.log(a.name);//name1
        console.log(b.name);//name2
        console.log(c.name);//name1
```

内置构造函数

JS里面一切皆为对象

https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects

1. Number: 数字

- new Number(100.456) ： 创建了一个数字的实例化对象；
- 实例化对象.toFixed()：保留多少位小数；

```
var num = new Number(100.456);//typeof  num  ====>  object
console.log(num); // Number{100.456}
console.log(num.toFixed(2)); //100.46

var num1 = 100.456;
console.log(num1); // typeof  num1  ====>  number  //100.456

JS解释器在执行JS代码的时候，
如果需要，会尝试把当前数据转换成他需要的数据类型 ，
这种转换我们称之为  隐式转换；

console.log(num1.toFixed(2)); //100.46

/*小数转整数*/
var num1 = 100.456;
console.log(parseInt(num1));//parseInt 内置的全局方法，在window对象身上 //100
/* 为了规范，把所有的关于数字的操作方法全部集中到Number身上*/
console.log(Number.parseInt(num1));  // 静态方法：是通过构造函数直接调用的，参数一般就是这个构造函数的实例化对象

/* 把字符串转换成数字*/
var count = '120';
/* 显式的数据类型转换*/
console.log(typeof Number(count), Number(count)); //number 120
console.log(typeof Number.parseInt(count), Number.parseInt(count)); //number 120
/* 隐式的：用起来很方便*/
console.log(typeof (count/2), (count/2)); //number 60

 /* 布尔值转换为数字*/
console.log(Number(true), Number(false));  //1   0
```

- NaN：Not A Number

  类型是number类型；
  和任何值都不相等，包括他自己；
  参与的数字运算返回值始终是NaN；
  使用isNaN来判断一个变量是不是NaN；

```
var num1 = 'a100';
/** 如果转换失败，返回值是NaN*/
var r1 = Number.parseInt(num1);
console.log(r1);//NaN

/** NaN的类型是number*/
console.log(typeof r1);// number

 /* 和任何值都不相等，包括他自己*/
console.log(r1  ==  NaN);//false
/* 使用isNaN这个方法来判断是否是NaN*/
console.log(Number.isNaN(r1));//true
console.log(isNaN(NaN));//true
console.log(isNaN(100));//false
```

- 数字运算：加减乘除取余；
- 自运算：++ --

  --i or ++i：先运算，后执行；

  i-- or i++：先执行，后运算；

  在循环里面通常通过自运算去修改判断条件；

```
var n = 100, m = 200;
console.log(n + m);//300
console.log(n - m);//-100
console.log(n * m);//20000
console.log(n / m);//0.5
console.log(n % m); //取余数  100

/* 自运算*/
var i = 8;
/** --放在数字前面：先运算，后执行*/
console.log(--i); // 减去 1,然后打印 7

var j = 8;
/** --放在数字后面：先执行，后运算*/
console.log(j--); // 先打印，j:8，然后减去 1
console.log('运算之后的结果：', j); //  7
```

内置对象

JavaScript 提供了多个内置对象：Math、 Date 、Array、String等

2. 数学：Math

- 有各种关于数学的操作方法
- https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Math
- 随机数方法random():

  random() 方法可以随机返回一个小数，其取值范围是 [0，1)，左闭右开 0 <= x < 1

  得到一个两数之间的随机整数，包括第一个数，不包括第二个数

```
// Math数学对象，不是一个构造函数，所以我们不需要new 来调用，而是直接使用里面的属性和方法即可

Math.PI                 // 圆周率
Math.floor()            // 向下取整
Math.ceil()             // 向上取整
Math.round()            // 四舍五入版 就近取整   注意：5特殊，它往大了取  1.5结果是2  -3.5结果是  -3
Math.abs()              // 绝对值
Math.max()/Math.min() // 求最大和最小值

/*注意：Math数学对象使用时要括号*/
console.log(Math.PI);
console.log(Math.max(1,99,3)); // 99

/*随机数方法random()*/
//random() 方法可以随机返回一个小数，其取值范围是 [0，1)，左闭右开 0 <= x < 1
//得到一个两数之间的随机整数，包括第一个数，不包括第二个数

// 得到两个数之间的随机整数，并且包含这两个整数
function rand(min,max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}
console.log(rand(1,10));


```

For循环

通过指定的规则不断的去做某一件事情；

for(①(条件)初始化语句; ②条件判断语句; ③条件修改语句){ ④代码块 }

执行顺序：

- 第一步，执行条件初始化语句；
- 第二步，执行条件判断语句，看是true 还是 false；
- 第三步，如果是true，就执行代码块；
  代码块执行完成后，执行条件修改语句；
  条件修改语句执行完成后，进入第二步；
- 如果是false，循环结束；

```
for (var n = 10; n < 100; n++){
// var n = 10:（判断条件）初始化语句;
// n<100:条件判断语句，是true还是false;
//如果是true，就执行循环的代码块: i>e 是true ，执行console.log(i)
//如果是false，就结束循环;
     console.log(n*n);//循环代码
}
```

For...in：循环对象

```
var obj = {
    name: '小明',
    age: 18,
    sex: '男',
    fn:function() {};
};
console.log(obj.name);
console.log(obj.age);
console.log(obj.sex);

//for in 遍历我们的对象
//for (变量 in 对象){}
//for (变量 in 对象) 变量一般使用k或者key
for(var k in obj){ //var k用来存储遍历出来的属性
    console.log(k); // k 变量 输出得到的是属性名和方法名
    console.log(obj[k]); // obj[k] 得到的是属性值; k是个变量，要想拿到对应的属性值，需要使用[]。因为[]是支持变量的，不能使用.拿到属性值

}
```

条件判断语句：

最简单的条件判断语句：

If(判断条件){ 代码块1 }

如果判断条件为true，就执行代码块1，否则什么都不干。

```
/** 完整的条件判断语句  如果...否则*/
var i = 10;
if (i > 10) { //如果判断条件为true，就执行代码块
    console.log('i 大于 10');
} else if (i == 10) {
    console.log('i 等于 10');
} else { //else后面没有判断条件，如果上面的if后面的条件都没有成立，就执行该代码块
    console.log('i 小于 10');
}
```